EXHIBIT A

Andrew W. Appel, Curriculum Vitae

Andrew W. Appel
Eugene Higgins Professor of Computer Science
Department of Computer Science, Princeton University
35 Olden Street, Princeton NJ 08540

appel@princeton.edu, +1-609-258-4627, fax: +1-609-258-2016
https://www.cs.princeton.edu/~appel

Research Interests

Software verification, programming languages, computer security, compilers, semantics, software engineering, information technology policy, elections and voting technology.

Education

A.B. *summa cum laude* (physics) Princeton University, 1981
Ph.D. (computer science) Carnegie-Mellon University, 1985

Professional Appointments

Princeton University, Princeton, NJ. Eugene Higgins Professor of Computer Science, since 2011; Department Chair, 2009-15; Professor of Computer Science, since 1995; Associate Chair, 1997-2007; Assoc. Prof., 1992-95; Asst. Prof. 1986-92.

Massachusetts Institute of Technology. Visiting Professor, July-December 2013.

INRIA (Institut National de Recherche en Informatique et en Automatique), Rocquencourt, France. Visiting Professor, academic year 2005-06 & summers 2004, 2007.

Bell Laboratories, Murray Hill, NJ. Member of Technical Staff, Summer 1984. Consultant, 1983-2001.

Carnegie-Mellon University, Pittsburgh, PA. Research and teaching assistant, 1982-85.

College of Medicine, University of Illinois, Urbana, IL. Computer programmer, summers 1976-80.

Awards and Honors

Kusaka Memorial Prize in Physics, Princeton University, 1981.

National Science Foundation Graduate Student Fellowship, 1981-1984.

ACM Fellow (Association for Computing Machinery), 1998.

The Other Prize, Programming Contest of the ACM International Conference on Functional Programming, 1998.

ACM SIGPLAN Distinguished Service Award, 2002.

ACM SIGPLAN selected "Real-time Concurrent Collection on Stock Multiprocessors" (Appel, Ellis, Li 1988) as one of the 50 most influential papers in 20 years of the PLDI conference, 2002.

Professional Activities

1. Program Committee, *ACM SIGPLAN '89 Conf. on Prog. Lang. Design and Implementation*, 1989.
2. Program Committee, *Seventeenth ACM Symp. on Principles of Programming Languages*, 1990.
3. Associate Editor, *ACM Transactions on Programming Languages and Systems*, 1990-1992.
4. Associate Editor, *ACM Letters on Programming Languages and Systems*, 1991-1992.
5. Program Chair, *Nineteenth ACM Symp. on Principles of Programming Languages*, 1992.
6. Co-editor, *Journal of Functional Programming* special issue on ML, 1992.
7. Program Committee, *Sixth ACM Conf. on Functional Prog. Lang. and Computer Architecture*, 1993.
8. Editor in Chief, *ACM Transactions on Programming Languages and Systems*, 1993-97.
9. Program Committee, *International Conference on Functional Programming*, 1997.
10. General Chair, *POPL'99: 26th ACM Symp. on Principles of Programming Languages*, 1999.
11. Program Committee, *IEEE Symposium on Security and Privacy,* 2002.
12. Program Committee, *ACM SIGPLAN Workshop on Types in Language Design and Implementation,* 2003.
13. Program Committee, *Nineteenth Annual IEEE Symposium on Logic in Computer Science*, 2004.
14. Program Committee, *ACM SIGPLAN 2005 Conference on Programming Language Design and Implementation (PLDI)*, 2005.
15. Program Committee, *International Workshop on Logical Frameworks and Meta-Languages: Theory and Practice (LFMTP'06)*, 2006.
16. Program Committee, *EVT'07: 2007 Usenix/ACCURATE Electronic Voting Technology Workshop*.
17. Program Committee, *POPL'09: 36th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 2009.
18. Program Committee, *PLDI 2011: 32nd ACM SIGPLAN conference on Programming Language Design and Implementation*, 2011.
19. General Co-Chair, *ITP 2012: Interactive Theorem Proving*, 2012.
20. Program Committee, *POPL 2014: 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 2014.
21. Award Committee, SIGPLAN Programming Languages Software Award, 2016.
22. Board of Advisors, *Verified Voting Foundation*, since 2015.
23. Program Committee, *POPL 2020: 47th ACM SIGPLAN Symposium on Principles of Programming Languages*, 2020.

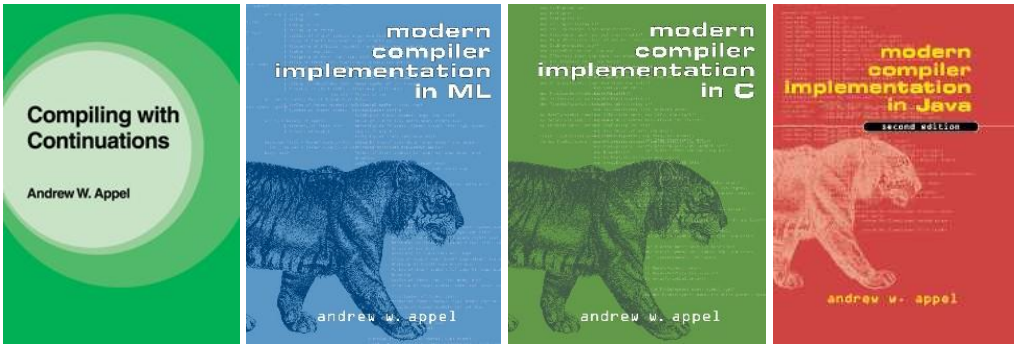
Research Grants

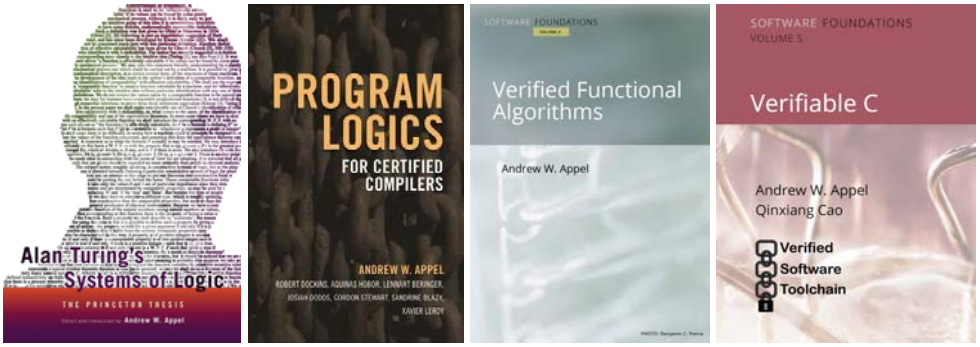
1. *Implementation of an efficient reducer for lambda expressions,* National Science Foundation DCR-8603453, $115,799, 1986-88.
2. Digital Equipment Corporation Faculty Incentive Grant, $180,000, 1986-89.
3. *Unifying compile-time and run-time evaluation,* National Science Foundation CCR-8806121, $123,510, 1988-90.

4. *Standard ML of New Jersey software capitalization,* National Science Foundation CCR-8914570, $119,545, 1990-91.
5. *Using immutable types for debugging and parallelism,* National Science Foundation CCR-9002786, $174,618, 1990-92.
6. *Optimization of space usage,* National Science Foundation CCR-9200790, $348,119, 1992-96.
7. *Framework, Algorithms, and Applications for Cross-module Inlining,* National Science Foundation CCR-9625413, $180,331, 1996-98.
8. *Development of a HIL/LIL Framework for a National Compiler Infrastructure*, Defense Advanced Research Projects Agency and National Science Foundation (as subcontractor to Univ. of Virginia), $1,397,293, 1996-99.
9. *Tools, Interfaces, and Access Control for Secure Programming,* National Science Foundation CCR-9870316, $322,000, 1998-2001 (co-PI).
10. *Scaling Proof-Carrying Code to Production Compilers and Security Policies*, Defense Advanced Research Projects Agency, $3,870,378, 1999-2004.
11. *Applying Compiler Techniques to Proof-Carrying Code*, National Science Foundation CCR-9974553, $220,000, 1999-2002.
12. IBM University Partnership Program, $40,000, 1999-2000.
13. *High-Assurance Common Language Runtime*, National Science Foundation CCR-0208601, $400,000, 2002-2005.
14. *Assurance-Carrying Components*, Advanced Research and Development Agency contract NBCHC030106, $759,910, 2003-05.
15. Sun Microsystems research grant, $20,000, 2004.
16. *End-to-end source-to-object verification of interface safety,* National Science Foundation grant CCF-0540914, $325,000, 2006-09.
17. *MulVAL Technologies Plan,* New Jersey Commission on Science and Technology, $60,000, 2006.
18. Microsoft Corporation research grant, $25,000, 2006.
19. *Evidence-based Trust in Large-scale MLS Systems*, Air Force Office of Scientific Research FA9550-09-1-0138 (as subcontractor to Kansas State University), $1,000,000, 2009-14.
20. *Combining Foundational and Lightweight Formal Methods to Build Certifiably Dependable Software*, National Science Foundation grant CNS-0910448, $500,000, 2009-13.
21. *CARS: A Platform for Scaling Formal Verification to Component-Based Vehicular Software Stacks*, Defense Advanced Research Projects Agency award FA8750-12-2-0293, $6,108,346, 2012-2017.
22. *Verified HMAC*, Google Advanced Technology and Projects grant, $95,928, 2014.
23. *Principled Optimizing Compilation of Dependently Typed Languages*, National Science Foundation grant CCF-1407794, $600,000, 2014-17.
24. *Concurrent separation logic for C*, Intel Corporation research grant, $238,015, 2015-16.
25. *Collaborative Research: Expeditions in Computing: The Science of Deep Specification,* National Science Foundation grant CCF-1521602, $3,453,419, 2015-20.

Publications

Books, chapters in books





1. ``Garbage Collection," in *Topics in Advanced Language Implementation,* Peter Lee, ed. MIT Press, 1991.
2. *Compiling with Continuations,* Cambridge University Press, 1992.
3. *Modern Compiler Implementation in ML,* Cambridge University Press, 1998.
4. *Modern Compiler Implementation in Java,* Cambridge University Press, 1998.
5. *Modern Compiler Implementation in C,* Cambridge University Press, 1998.
6. *Modern Compiler Implementation in Java, 2nd edition*, with Jens Palsberg, Cambridge University Press, 2002.
7. *Alan Turing's Systems of Logic: The Princeton Thesis*, edited and introduced by Andrew W. Appel, Princeton University Press, 2012.
8. *Program Logics for Certified Compilers*, by Andrew W. Appel with Robert Dockins, Aquinas Hobor, Lennart Beringer, Josiah Dodds, Gordon Stewart, Sandrine Blazy, and Xavier Leroy. Cambridge University Press, 2014.
9. *Verified Functional Algorithms*, by Andrew W. Appel, 2017. Volume 3 of *Software Foundations*, edited by B. C. Pierce.
10. *Verifiable C*, by Andrew W. Appel, 2020. Volume 5 of *Software Foundations*, edited by B. C. Pierce.

Journal papers, refereed conference papers, and patents

11. A Microprocessor-Based CAI System with Graphic Capabilities, by Frank J. Mabry, Allan H. Levy, and Andrew W. Appel, *Proc. 1978 conference, Assoc. for Development of Computer-based Instruction Systems.*
12. Rogomatic: A Belligerent Expert System, by Michael L. Mauldin, Guy J. Jacobson, Andrew W. Appel, and Leonard G. C. Hamey. *Proc. Fifth Nat. Conf. Canadian Soc. for Computational Studies of Intelligence*, May 1984.
13. An Efficient Program for Many-Body Simulations. *SIAM Journal on Scientific and Statistical Computing* 6(1):85-103, 1985.
14. Semantics-Directed Code Generation, by Andrew W. Appel, *Proc. Twelfth ACM Symposium on Principles of Programming Languages,* January 1985.

15. Generalizations of the Sethi-Ullman algorithm for register allocation. Andrew W. Appel and Kenneth J. Supowit, *Software \(em Practice and Experience* 17(6):417-421, 1987.
16. A Standard ML compiler, by Andrew W. Appel and David B. MacQueen, *Proc. Third Int'l Conf. on Functional Programming & Computer Architecture (LNCS 274, Springer-Verlag),* Portland, Oregon, September 1987.
17. Garbage collection can be faster than stack allocation. Andrew W. Appel. *Information Processing Letters* 25(4):275-279, 17 June 1987.
18. Real-time concurrent collection on stock multiprocessors, by Andrew W. Appel, John Ellis, and Kai Li, *Proc. ACM SIGPLAN '88 Conf. on Prog. Lang. Design & Implementation,* pp. 11-20, June 1988.
19. The World's Fastest Scrabble Program. Andrew W. Appel and Guy J. Jacobson, *Comm. ACM* 31(5):572-578,585, May 1988.
20. Simulating digital circuits with one bit per wire. Andrew W. Appel, *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* 7(9):987-993, September 1988.
21. Continuation-passing, closure-passing style, by Andrew W. Appel and Trevor Jim, *Proc. Sixteenth ACM Symposium on Principles of Programming Languages,* pp. 293-302, January 1989.
22. Simple Generational Garbage Collection and Fast Allocation. Andrew W. Appel. *Software--Practice and Experience* 19(2):171-183, February 1989.
23. Allocation without Locking. Andrew W. Appel. *Software--Practice and Experience* 19(7):703-705, July 1989.
24. Runtime Tags Aren't Necessary. Andrew W. Appel. *Lisp and Symbolic Computation* 2, 153-162 (1989).
25. Vectorized Garbage Collection. Andrew W. Appel and Aage Bendiksen. *The Journal of Supercomputing* 3, 151-160 (1989).
26. A Runtime System. *Lisp and Symbolic Computation* 3, 343-380, 1990.
27. An advisor for flexible working sets, by Rafael Alonso and Andrew W. Appel, *1990 ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems,* pp. 153-162, May 1990.
28. Debugging Standard ML without reverse engineering, by Andrew P. Tolmach and Andrew W. Appel, *Proc. 1990 ACM Conf. on Lisp and Functional Programming,* pp. 1-12, June 1990.
29. Real-time concurrent garbage collection system and method, by John R. Ellis, Kai Li, and Andrew W. Appel. U.S. Patent 5,088,036, 1992.
30. Virtual memory primitives for user programs, by Andrew W. Appel and Kai Li, *Proc. Fourth Int'l Conf. on Architectural Support for Prog. Languages and Operating Systems,* (SIGPLAN Notices 26(4)) pp. 96-107, April 1991.
31. Standard ML of New Jersey, by Andrew W. Appel and David B. MacQueen, *Third Int'l Symp. on Prog. Lang. Implementation and Logic Programming, Springer-Verlag LNCS 528,* pp. 1-13, August 1991.
32. Callee-save registers in Continuation-Passing Style, by Andrew W. Appel and Zhong Shao. *Lisp and Symbolic Computation* 5, 189-219, 1992.
33. Smartest Recompilation, by Zhong Shao and Andrew W. Appel, *Proc. Twenthieth ACM Symp. on Principles of Programming Languages,* January 1993.
34. A Critique of Standard ML. Andrew W. Appel. *Journal of Functional Programming* 3 (4) 391-430, 1993.
35. Unrolling Lists, by Zhong Shao, John H. Reppy, and Andrew W. Appel, *Proc. 1994 ACM Conf. on Lisp and Functional Programming,* pp. 185-195, June 1994.
36. Space-Efficient Closure Representations, by Zhong Shao and Andrew W. Appel, *Proc. 1994 ACM Conf. on Lisp and Functional Programming,* pp. 150-161, June 1994.
37. Separate Compilation for Standard ML, by Andrew W. Appel and David B. MacQueen, *Proc. 1994 ACM Conf. on Programming Language Design and Implementation* (SIGPLAN Notices v. 29 #6), pp. 13-23, June 1994.
38. Axiomatic Bootstrapping: A guide for compiler hackers, Andrew W. Appel, *ACM Transactions on Programming Languages and Systems*, vol. 16, number 6, pp. 1699-1718, November 1994.

39. Loop Headers in Lambda-calculus or CPS. Andrew W. Appel. *Lisp and Symbolic Computation* 7, 337-343, 1994.
40. A Debugger for Standard ML. Andrew Tolmach and Andrew W. Appel. *Journal of Functional Programming*, vol. 5, number 2, pp. 155-200, April 1995.
41. A Type-Based Compiler for Standard ML, by Zhong Shao and Andrew W. Appel, *Proc. 1995 ACM Conf. on Programming Language Design and Implementation* (SIGPLAN Notices v. 30 #6), pp. 116-129, June 1995.
42. Cache Performance of Fast-Allocating Programs, by Marcelo J. R. Goncalves and Andrew W. Appel, *Proc. Seventh Int'l Conf. on Functional Programming and Computer Architecture,* pp. 293-305, ACM Press, June 1995.
43. Empirical and Analytic Study of Stack versus Heap Cost for Languages with Closures. Andrew W. Appel and Zhong Shao. *Journal of Functional Programming* 6 (1) 47-74, 1996.
44. How to Edit a Journal by E-mail. Andrew W. Appel *Journal of Scholarly Publishing* 27 (2) 82-99, January 1996.
45. Iterated Register Coalescing, by Lal George and Andrew W. Appel, *23rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* pp. 208-218, January 1996.
46. Iterated Register Coalescing. Lal George and Andrew W. Appel. *ACM Transactions on Programming Languages and Systems* 18(3) 300-324, May 1996. Shorter version appeared in *23rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, January 1996.
47. Security and document compatibility for electronic refereeing. Andrew W. Appel. *CBE Views* 20(1), 1997, published by the Council of Biology Editors.
48. Lambda-Splitting: A Higher-Order Approach to Cross-Module Optimizations, by Matthias Blume and Andrew W. Appel, *Proc. ACM SIGPLAN International Conference on Functional Programming (ICFP '97)*, pp. 112-124, June 1997.
49. The Zephyr Abstract Syntax Description Language, by Daniel C. Wang, Andrew W. Appel, Jeff L. Korn, and Christopher S. Serra. *Conference on Domain-Specific Languages*, USENIX Association, October 1997.
50. Shrinking Lambda Expressions in Linear Time. Andrew W. Appel and Trevor Jim. *Journal of Functional Programming* v. 7 no. 5, pp. 515-540, 1997.
51. Traversal-based Visualization of Data Structures, by Jeffrey L. Korn and Andrew W. Appel, *IEEE Symposium on Information Visualization (InfoVis '98)*, pp. 11-18, October 1998.
52. Hierarchical Modularity. Matthias Blume and Andrew W. Appel, *ACM Transactions on Programming Languages and Systems*, 21 (4) 812-846, July 1999.
53. Lightweight Lemmas in Lambda Prolog, by Andrew W. Appel and Amy Felty, *16th International Conference on Logic Programming*, pp. 411-425, MIT Press, November 1999.
54. Proof-Carrying Authentication, by Andrew W. Appel and Edward Felten, *6th ACM Conference on Computer and Communications Security*, November 1999.
55. Efficient and Safe-for-Space Closure Conversion, Zhong Shao and Andrew W. Appel, *ACM Trans. on Prog. Lang. and Systems* 22(1) 129-161, January 2000.
56. A Semantic Model of Types and Machine Instructions for Proof-Carrying Code, by Andrew W. Appel and Amy P. Felty. *27th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '00)*, pp. 243-253, January 2000.
57. Machine Instruction Syntax and Semantics in Higher Order Logic, by Neophytos G. Michael and Andrew W. Appel, *17th International Conference on Automated Deduction (CADE-17)*, Springer-Verlag (Lecture Notes in Artificial Intelligence), pp. 7-24, June 2000.
58. Technological Access Control Interferes with Noninfringing Scholarship. Andrew W. Appel and Edward W. Felten. *Communications of the ACM* 43 (9) 21-23, September 2000.
59. An Indexed Model of Recursive Types for Foundational Proof-Carrying Code. Andrew W. Appel and David McAllester. *ACM Transactions on Programming Languages and Systems* 23 (5) 657-683, September 2001.

60. Type-Preserving Garbage Collectors, Daniel C. Wang and Andrew W. Appel, *POPL 2001: The 28th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pp. 166-178, January 2001.
61. SAFKASI: A Security Mechanism for Language-Based Systems, Dan S. Wallach, Andrew W. Appel, and Edward W. Felten. ACM Transactions on Software Engineering and Methodology, 9 (4) 341-378, October 2000.
62. Optimal Spilling for CISC Machines with Few Registers, by Andrew W. Appel and Lal George. *ACM SIGPLAN 2001 Conference on Programming Language Design and Implementation* , pp. 243-253, June 2001.
63. Foundational Proof-Carrying Code, by Andrew W. Appel, *16th Annual IEEE Symposium on Logic in Computer Science* (LICS '01), pp. 247-258, June 2001.
64. A Stratified Semantics of General References Embeddable in Higher-Order Logic, by Amal Ahmed, Andrew W. Appel, and Roberto Virga. *17th Annual IEEE Symposium on Logic in Computer Science (LICS 2002),* pp. 75-86, June 2002.
65. Creating and Preserving Locality of Java Applications at Allocation and Garbage Collection Times, by Yefim Shuf, Manish Gupta, Hubertus Franke, Andrew W. Appel, and Jaswinder Pal Singh. *17th Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2002), SIGPLAN Notices* 37(11) pp. 13-25, November 2002.
66. Mechanisms for secure modular programming in Java, by Lujo Bauer, Andrew W. Appel, and Edward W. Felten. *Software--Practice and Experience* 33:461-480, 2003.
67. A Trustworthy Proof Checker, by Andrew W. Appel, Neophytos G. Michael, Aaron Stump, and Roberto Virga. *Journal of Automated Reasoning* 31:231-260, 2003.
68. Using Memory Errors to Attack a Virtual Machine, by Sudhakar Govindavajhala and Andrew W. Appel, *2003 IEEE Symposium on Security and Privacy,* pp. 154-165, May 2003.
69. A Provably Sound TAL for Back-end Optimization, by Juan Chen, Dinghao Wu, Andrew W. Appel, and Hai Fang. *PLDI 2003: ACM SIGPLAN Conference on Programming Language Design and Implementation,* pp. 208-219, June 2003.
70. Foundational Proof Checkers with Small Witnesses, by Dinghao Wu, Andrew W. Appel, and Aaron Stump. *5th ACM-SIGPLAN International Conference on Principles and Practice of Declarative Programming*, pp. 264-274, August 2003.
71. Policy-Enforced Linking of Untrusted Components (Extended Abstract), by Eunyoung Lee and Andrew W. Appel, *European Software Engineering Conference* and *ACM SIGSOFT Symposium on the Foundations of Software Engineering*, pp. 371-374, September 2003.
72. Polymorphic Lemmas and Definitions in Lambda Prolog and Twelf, by Andrew W. Appel and Amy P. Felty. *Theory and Practice of Logic Programming* 4 (1) 1-39, January 2004.
73. Dependent Types Ensure Partial Correctness of Theorem Provers, by Andrew W. Appel and Amy P. Felty. *Journal of Functional Programming* 14(1):3-19, January 2004.
74. Construction of a Semantic Model for a Typed Assembly Language, by Gang Tan, Andrew W. Appel, Kedar N. Swadi, and Dinghao Wu. In *5th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI '04),* January 2004.
75. MulVAL: A Logic-based Network Security Analyzer by Xinming Ou, Sudhakar Govindavajhala, and Andrew W. Appel, In *14th Usenix Security Symposium,* August 2005.
76. A Compositional Logic for Control Flow by Gang Tan and Andrew W. Appel, in *7th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI)*, January 2006.
77. Safe Java Native Interface, by Gang Tan, Andrew W. Appel, Srimat Chakradhar, Anand Raghunathan, Srivaths Ravi, and Daniel Wang. *International Symposium on Secure Software Engineering,* March 2006.
78. A Very Modal Model of a Modern, Major, General Type System, by Andrew W. Appel, Paul-Andre Mellies, Christopher D. Richards, and Jerome Vouillon. *POPL 2007: The 34th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, January 2007.

79. Separation Logic for Small-step C minor, by Andrew W. Appel and Sandrine Blazy, in *TPHOLs 2007: 20th International Conference on Theorem Proving in Higher-Order Logics,* pp. 5-21, September 2007.
80. Oracle Semantics for Concurrent Separation Logic, by Aquinas Hobor, Andrew W. Appel, and Francesco Zappa Nardelli, in *ESOP'08: European Symposium on Programming,* April 2008.
81. Multimodal Separation Logic for Reasoning About Operational Semantics, by Robert Dockins, Andrew W. Appel, and Aquinas Hobor, in *Twenty-fourth Conference on the Mathematical Foundations of Programming Semantics*, May 2008.
82. The New Jersey Voting-machine Lawsuit and the AVC Advantage DRE Voting Machine, by Andrew W. Appel, Maia Ginsburg, Harri Hursti, Brian W. Kernighan, Christopher D. Richards, Gang Tan, and Penny Venetis. In *EVT/WOTE'09, 2009 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections*, August 2009.
83. A Fresh Look at Separation Algebras and Share Accounting by Robert Dockins, Aquinas Hobor, and Andrew W. Appel. *Seventh Asian Symposium on Programming Languages and Systems (APLAS 2009),* December 2009.
84. A Theory of Indirection via Approximation, by Aquinas Hobor, Robert Dockins, and Andrew W. Appel. *POPL 2010: The 37th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pp. 171-184, January 2010.
85. Formal Verification of Coalescing Graph-Coloring Register Allocation, by Sandrine Blazy, Benoit Robillard and Andrew W. Appel. *ESOP 2010: 19th European Symposium on Programming,* pp. 145-164, March 2010.
86. Concurrent Separation Logic for Pipelined Parallelization, by Christian J. Bell, Andrew W. Appel, and David Walker. In *SAS 2010: 17th Annual Static Analysis Symposium*, September 2010.
87. Semantic Foundations for Typed Assembly Languages, by A. Ahmed, A. W. Appel, C. D. Richards, K. Swadi, G. Tan, and D. C. Wang. *ACM Transactions on Programming Languages and Systems*, 32(3):7.1-7.67, March 2010.
88. A Logical Mix of Approximation and Separation by Aquinas Hobor, Robert Dockins, and Andrew W. Appel. In APLAS 2010: 8th ASIAN Symposium on Programming Languages and Systems, November 2010.
89. Local Actions for a Curry-style Operational Semantics by Gordon Stewart and Andrew W. Appel. In *PLPV'11: 5th ACM SIGPLAN Workshop on Programming Languages meets Program Verification*, January 29, 2011.
90. Verified Software Toolchain, by Andrew W. Appel. In *ESOP 2011: 20th European Symposium on Programming,* LNCS 6602, pp. 1-17, March 2011.
91. VeriSmall: Verified Smallfoot Shape Analysis, by Andrew W. Appel. In *CPP 2011: First International Conference on Certified Programs and Proofs, Springer LNCS 7086,* pp. 231-246, December 2011.
92. A Certificate Infrastructure for Machine-Checked Proofs of Conditional Information Flow, by Torben Amtoft, Josiah Dodds, Zhi Zhang, Andrew Appel, Lennart Beringer, John Hatcliff, Xinming Ou and Andrew Cousino. *First Conference on Principles of Security and Trust (POST 2012)*, LNCS 7215, pp. 369-389, March 2012.
93. A list-machine benchmark for mechanized metatheory by Andrew W. Appel, Robert Dockins, and Xavier Leroy. *Journal of Automated Reasoning* 49(3):453-491, 2012. DOI 10.1007/s10817-011-9226-1
94. Security Seals On Voting Machines: A Case Study, by Andrew W. Appel. *ACM Transactions on Information and System Security (TISSEC)* 14 (2) pages 18:1--18:29, September 2011.
95. Verified Heap Theorem Prover by Paramodulation, by Gordon Stewart, Lennart Beringer, and Andrew W. Appel. In *ICFP 2012: The 17th ACM SIGPLAN International Conference on Functional Programming*, pp. 3-14, September 2012.
96. Mostly Sound Type System Improves a Foundational Program Verifier, by Josiah Dodds and Andrew W. Appel. *3rd International Conference on Certified Programs and Proofs (CPP 2013),* December

2013.

97. Verified Compilation for Shared-memory C, by Lennart Beringer, Gordon Stewart, Robert Dockins, and Andrew W. Appel. *ESOP'14: 23rd European Symposium on Programming*, April 2014.
98. Portable Software Fault Isolation, by Joshua A. Kroll, Gordon Stewart, and Andrew W. Appel. *CSF'14: Computer Security Foundations Symposium*, IEEE Press, July 2014.
99. Compositional CompCert, by Gordon Stewart, Lennart Beringer, Santiago Cuellar, and Andrew W. Appel. *POPL 2015: The 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 275-287, January 2015.
100. Verified Correctness and Security of OpenSSL HMAC, by Lennart Beringer, Adam Petcher, Katherine Q. Ye, and Andrew W. Appel. In *24th USENIX Security Symposium,* pages 207-221, August 2015.
101. Verification of a Cryptographic Primitive: SHA-256, by Andrew W. Appel. *ACM Transactions on Programming Languages and Systems*, 37(2) 7:1-7:31, April 2015.
102. Modular Verification for Computer Security, by Andrew W. Appel, in *29th IEEE Computer Security Foundations Symposium (CSF'16),* June 2016.
103. Shrink Fast Correctly! by Olivier Savary Belanger and Andrew W. Appel. *Proceedings of International Symposium on Principles and Practice of Declarative Programming (PPDP'17)*, 12 pages, October 2017 (PPDP'17).
104. Verified Correctness and Security of mbedTLS HMAC-DRBG by Katherine Q. Ye, Matthew Green, Naphat Sanguansin, Lennart Beringer, Adam Petcher, and Andrew W. Appel. *CCS'17: ACM Conference on Computer and Communications Security,* October 2017.
105. Bringing order to the separation logic jungle, by Qinxiang Cao, Santiago Cuellar, and Andrew W. Appel. *APLAS'17: 15th Asian Symposium on Programming Languages and Systems*, November 2017.
106. A verified messaging system, by William Mansky, Andrew W. Appel, and Aleksey Nogin. *OOPSLA'17: ACM Conference on Object-Oriented Programming Systems, Languages, and Applications*, October 2017. *Proceedings of the ACM on Programming Languages (PACM/PL)* volume 1, issue OOPSLA, paper 87, 2017.
107. Position paper: the science of deep specification, by Andrew W. Appel, Lennart Beringer, Adam Chlipala, Benjamin C. Pierce, Zhong Shao, Stephanie Weirich and Steve Zdancewic, *Philosophical Transactions of the Royal Society A* 375:21060331 (24 pages), 2017.
108. VST-Floyd: A separation logic tool to verify correctness of C programs, by Qinxiang Cao, Lennart Beringer, Samuel Gruetter, Josiah Dodds, and Andrew W. Appel. *Journal of Automated Reasoning* 61(1), pp. 367-422, 2018. (Local copy)
109. Closure Conversion is Safe for Space, by Zoe Paraskevopoulou and Andrew W. Appel. *Proceedings of the ACM on Programming Languages*, vol. 3, no. ICFP, article 83, 29 pages, doi 10.1145/3341687, August 2019.
110. Abstraction and Subsumption in Modular Verification of C Programs, by Lennart Beringer and Andrew W. Appel. *FM2019: 23rd International Symposium on Formal Methods*, October 2019.
111. Connecting Higher-Order Separation Logic to a First-Order Outside World, by William Mansky, Wolf Honoré, and Andrew W. Appel, *ESOP 2020: European Symposium on Programming*, April 2020.
112. Ballot-Marking Devices (BMDs) Cannot Assure the Will of the Voters, by Andrew W. Appel, Richard A. DeMillo, and Philip B. Stark. To appear in *Election Law Journal*, 2020. (Earlier versions appeared on SSRN.)
113. Verified sequential malloc/free, by Andrew W. Appel and David A. Naumann, in *2020 ACM SIGPLAN International Symposium on Memory Management,* June 2020.

Workshop and unrefereed conference papers

114. Debuggable concurrency extensions for Standard ML, by Andrew P. Tolmach and Andrew W. Appel, *Proc. ACM/ONR Workshop on Parallel and Distributed Debugging,* May 1991 (SIGPLAN Notices,

Dec. 1991), pp. 115-127.

115. Efficient Substitution in Hoare Logic Expressions, by Andrew W. Appel, Kedar Swadi, and Roberto Virga. *4th International Workshop on Higher-Order Operational Techniques in Semantics (HOOTS 2000)*, pp. 35-50, September 2000.
116. Fair use, public domain, or piracy ... should the digital exchange of copyrighted works be permitted or prevented? (Rountable Panel II: Digital Video), by Andrew W. Appel, Jeffrey Cunard, Martin Garbus, and Edward Hernstadt, *Fordham Intellectual Property, Media & Entertainment Law Journal*, volume 11, number 2, page 317, 2001.
117. A Trustworthy Proof Checker, by Andrew W. Appel, Neophytos G. Michael, Aaron Stump, and Roberto Virga. In *Verification Workshop - VERIFY 2002* and (jointly) in *Foundations of Computer Security - FCS 2002* Copenhagen, Denmark, July 25-26, 2002.
118. A list-machine benchmark for mechanized metatheory (extended abstract) by Andrew W. Appel and Xavier Leroy. *LFMTP'06: International Workshop on Logical Frameworks and Meta-Languages: Theory and Practice*, August 2006.
119. Effective Audit Policy for Voter-Verified Paper Ballots, presented at *2007 Annual Meeting of the American Political Science Association*, Chicago, September 1, 2007.

Review Articles, Tutorials, Position Papers

120. Book Review of *Garbage Collection: Algorithms for Automatic Dynamic Memory Management* by Richard Jones and Rafael Lins. *Journal of Functional Programming* 7(2), pp. 227-229, March 1997.
121. SSA is Functional Programming. *ACM SIGPLAN Notices* v. 33, no. 4, pp. 17-20, April 1998.
122. Protection against untrusted code. *IBM Developer Works*, September 1999.
123. Retrospective: Real-time Concurrent Collection on Stock Multiprocessors. *20 Years of the ACM/SIGPLAN Conference on Programming Language Design and Implementation (1979-1999): A Selection,* ACM Press, 2004.
124. Foundational High-level Static Analysis. In *CAV 2008 Workshop on Exploiting Concurrency Efficiently and Correctly*, July 2008.
125. Technical Perspective: The Scalability of CertiKOS, by Andrew W. Appel, *Communications of the ACM*, vol. 62 no.10, page 88. DOI 10.1145/335690610.1145/3356906.
126. *Freedom-to-Tinker:* 16 articles on the freedom-to-tinker.com blog between 2007 and 2009; 6 articles in 2010; 15 articles in 2011.
127. The Birth of Computer Science at Princeton in the 1930s, in A. W. Appel, ed., *Alan Turing's Systems of Logic: The Princeton Thesis*, Princeton University Press, 2012.
128. Research Needs for Secure, Trustworthy, and Reliable Semiconductors, by Andrew Appel, Chris Daverse, Kenneth Hines, Rafic Makki, Keith Marzullo, Celia Merzbacher, Ron Perez, Fred Schneider, Mani Soma, and Yervant Zorian. Final workshop report of the NSF/CCC/SRC workshop on Convergence of Software Assurance Methodologies and Trustworthy Semiconductor Design and Manufacture, 2013.
129. CertiCoq: A verified compiler for Coq, by Abhishek Anand, Andrew Appel, Greg Morrisett, Zoe Paraskevopoulou, Randy Pollack, Olivier Savary Belanger, Matthieu Sozeau, and Matthew Weaver. In *CoqPL'17: The Third International Workshop on Coq for Programming Languages*, January 2017.
130. Position paper: the science of deep specification, by Andrew W. Appel, Lennart Beringer, Adam Chlipala, Benjamin C. Pierce, Zhong Shao, Stephanie Weirich, Steve Zdancewic. *Philosophical Transactions of the Royal Society A* vol. 375, no. 2104, September 2017.
131. *Securing the Vote: Protecting American Democracy*, by National Academies of Science, Engineering, and Medicine: Lee C. Bollinger, Michael A. McRobbie, Andrew W. Appel, Josh Benaloh, Karen Cook, Dana DeBeauvoir, Moon Duchin, Juan E. Gilbert, Susan L. Graham, Neal Kelley, Kevin J. Kennedy, Nathaniel Persily, Ronald L. Rivest, Charles Stewart III. September 2018.

132. Evidence-Based Elections: Create a Meaningful Paper Trail, then Audit, by Andrew W. Appel and Philip B. Stark, *Georgetown Law Technology Review*, volume 4, pages 523-541, 2020.

Unrefereed papers

133. An Investigation of Galaxy Clustering Using an Asymptotically Fast N-Body Algorithm. Senior Thesis, Princeton University, 1981.
134. Compile-time Evaluation and Code Generation in Semantics-Directed Compilers. Ph.D. Thesis, Carnegie-Mellon University, July 1985.
135. Concise specifications of locally optimal code generators, Princeton Univ. Dept. of Computer Science CS-TR-080-87, 1987.
136. Re-opening closures, Princeton Univ. Dept. of Computer Science CS-TR-079-87, February 1987.
137. Optimizing closure environment representations, by Andrew W. Appel and Trevor Jim. Princeton Univ. Dept. of Computer Science CS-TR-168-88, July 1988.
138. Unifying Exceptions with Constructors in Standard ML, with David MacQueen, Robin Milner, and Mads Tofte. Univ. of Edinburgh Dept. of Comp. Sci. CSR-266-88, May 1988.
139. Profiling in the presence of optimization and garbage collection, by Andrew W. Appel, Bruce Duba, and David MacQueen. CS-TR-197-88, November 1988.
140. Hash-Consing Garbage Collection, by Andrew W. Appel and Marcelo J.R. Goncalves, Technical report TR-412-93, Department of Computer Science, Princeton University, January 1993.
141. Emulating Write-Allocate on a No-Write-Allocate Cache, by Andrew W. Appel, CS-TR-459-94, Princeton University, June 20, 1994.
142. Is POPL Mathematics or Science?, by Andrew W. Appel, *ACM SIGPLAN Notices* 27 (4), pp. 87-89, April 1992.
143. Intensional Equality ;=) for Continuations, by Andrew W. Appel, *ACM SIGPLAN Notices* 31 (2), pp. 55-57, February 1996.
144. Ceci n'est pas une urne: On the Internet vote for the *Assemblée des Français de l'Etranger*, by Andrew W. Appel, June 2006.
145. Insecurities and Inaccuracies of the Sequoia AVC Advantage 9.00H DRE Voting Machine, by Andrew W. Appel, Maia Ginsburg, Harri Hursti, Brian W. Kernighan, Christopher D. Richards, and Gang Tan. October 2008.
146. The CompCert Memory Model, Version 2, by Xavier Leroy, Andrew W. Appel, Sandrine Blazy, and Gordon Stewart. INRIA Research Report RR-7987, June 2012.
147. Compiler Correctness for Concurrency: from concurrent separation logic to shared-memory assembly language, by Santiago Cuellar, Nick Giannarakis, Jean-Marie Madiot, William Mansky, Lennart Beringer, and Andrew W. Appel, Technical report TR-014-19, Department of Computer Science, Princeton University, March 2020.
148. Fair Elections During a Crisis: Urgent Recommendations in Law, Media, Politics, and Tech to Advance the Legitimacy of, and the Public Confidence in, the November 2020 U.S. Elections.,by the Ad Hoc Committee for 2020 Election Fairness and Legitimacy (Appel, Azari, Cain, *et al.*), edited by Richard L. Hasen, UCI Law School, April 2020.

PhD Students

1. Andrew P. Tolmach, Ph.D. (1992) *Debugging Standard ML*. Professor, Portland State University.
2. Zhong Shao, Ph.D. (1994) *Compiling Standard ML for Efficient Execution on Modern Machines*. Professor, Yale University.
3. Marcelo J. R. Goncalves, Ph.D. (1995) *Cache Performance of Programs with Intensive Heap Allocation and Generational Garbage Collection*.

4. Matthias Blume, Ph.D. (1997) *Hierarchical Modularity and Intermodule Optimization*. Computer Scientist, Google, Inc.
5. Richard (Drew) Dean, Ph.D. (1999) *Formal Aspects of Mobile Code Security*. Senior Computer Scientist, SRI International.
6. Jeffrey L. Korn, Ph.D. (1999) *Abstraction and Visualization in Graphical Debuggers*. Software Engineer, Google, Inc.
7. Daniel C. Wang, Ph.D. (2002) *Managing Memory with Types.* Computer Scientist, Amazon.com.
8. Kedar N. Swadi, Ph.D. (2003) *Typed Machine Language*. CTO, AlgoAnalytics, Pune, India.
9. Lujo Bauer, Ph.D. (2003) *Access Control for the Web via Proof-Carrying Authorization*. Associate Professor, Carnegie Mellon University.
10. Eunyoung Lee, Ph.D. (2003) *Secure Linking: A Logical Framework for Policy-Enforced Component Composition.* Associate Professor, Dongduk Women's University, Seoul, Korea.
11. Juan Chen, Ph.D. (2004) *A Low-Level Typed Assembly Language with a Machine-checkable Soundness Proof.* Computer Scientist, Google, Inc.
12. Amal J. Ahmed, Ph.D. (2004) *Semantics of Types for Mutable State*. Associate Professor, Northeastern University.
13. Gang Tan, Ph.D. (2005) *A Compositional Logic for Control Flow and its Application to Foundational Proof-Carrying Code*. Professor, Pennsylvania State University.
14. Dinghao Wu, Ph.D. (2005) *Interfacing Compilers, Proof Checkers, and Proofs for Foundational Proof-Carrying Code*. Associate Professor, Pennsylvania State University.
15. Xinming Ou, Ph.D. (2005) *A Logic Programming Approach to Network Security Analysis*. Professor, University of South Florida.
16. Sudhakar Govindavajhala, Ph.D. (2006) *A Formal Approach to Practical Network Security Management*. Computer and network security consultant.
17. Aquinas Hobor, Ph.D. (2008) *Oracle Semantics*. Assistant Professor, National University of Singapore and Yale/NUS college.
18. Christopher D. Richards, Ph.D. (2010) *The Approximation Modality in Models of Higher-Order Types*. Computer Scientist, Google, Inc.
19. Robert Dockins, Ph.D. (2012) *Operational Refinement for Compiler Correctness.* Researcher, Galois.com.
20. James Gordon Stewart, Ph.D. (2015) *Verified Separate Compilation for C.* Assistant Professor, Ohio University.
21. Josiah Dodds, Ph.D. (2015) *Computation Improves Interactive Symbolic Execution.* Researcher, Galois.com.
22. Qinxiang Cao, Ph.D. (2018) *Separation-Logic-based Program Verification in Coq.* Assistant Professor, Shanghai Jiao Tong University.
23. Olivier Savary Bélanger, Ph.D. (2019) *Verified Extraction for Coq*. Researcher, Galois.com.
24. Santiago Cuellar, PhD (2020) *Concurrent Permission Machine for modular proofs of optimizing compilers with shared memory concurrency*. Researcher, Galois.com.